Файл(стандартные функции языка)

Язык Си "рассматривает" файл как структуру. В stdio.h содержится определение структуры файла. Определен шаблон и директива препроцессора #define FILE struct iobuf

FILE тип переменной, которая является в дальнейшей работе дескриптором файла. Некоторые системы используют директиву typedef для установления этого соответствия.

Открытие файла fopen()

Функцией управляют три параметра.

*FILE *in;* //указатель потока

Для связывания указателя с файлом служит функция открытия файла fopen(), которая объявлена в заголовочном файле <stdio.h>.

in = fopen("test", "r");

1 параметр - имя открываемого файла

2 параметр -

"r"-по чтению "r+"-чтение и запись

"w"-по записи "w+"-запись и чтение, если файл уже был, он перезаписываетя

"a"-дозапись "a+"-чтение и дозапись, если файла еще не было, он создается

"b"-двоичный файл

"t"-текстовый файл

in является указателем на файл "test". Теперь будем обращаться к файлу через этот указатель.

Если файл не был открыт (его нет, нет места на диске), то возвращается в указатель 0.

```
if((in=fopen("test", "r"))==0)
    puts("Ошибка открытия файла");
```

Можно по другому

```
in=fopen("test", "r");
if (!in)
    puts("Ошибка открытия файла");
```

Закрытие файла fclose()

fclose(FILE *stream); //Если файл закрыт успешно, то возвращается 0 иначе -1.

Функции ввода/вывода одного символа fgetc(), fputc()

1. Чтение из файла посимвольно

int fgetc(FILE *stream);

2. Запись в файл посимвольно

int fputc(int c, FILE *stream);

Код этого же символа и возвращается.

```
ch=fgetc(in);
fputc(ch,out);

# include <stadio.h>
void main(void){
   FILE *in,*out;
   char ch;
   if((in=fopen("prog1", "r"))==0)
      fputs("Ошибка открытия prog1");
   if((out=fopen("prog2", "w"))==0)
      fputs("Ошибка открытия prog2);
   while((ch=getc(in))!=EOF)              //"End Of File" константа определенная в
dos.h
      fputc(ch, out);
   fclose(in);
   fclose(out);
  }
```

Функции форматированного ввода/вывода в файл

1. Форматированный вывод в текстовый файл

int fprintf(FILE *stream,"управл.строка",arg1,…)

Возвращает количество записанных байтов.

2. Форматированный ввод из текстового файла

int fscanf(FILE *stream,"управл.строка",&arg1,…)

Возвращает количество прочитанных байтов.

```
# include<stadio.h>
void main(void){
FILE*in;
int age;
      in=fopen("prog1", "r");
fscanf(in,"%d",&age);
fclose(in);
      in=fopen("prog2", "w");
      fprintf(in,"число %d\n",age);
      fclose(in);
}
```

Модификаторы и спецификаторы, те же, что и в printf().

Функции ввода/вывода строки символов в файл

1. Чтение текстовой строки из файла.

char* fgets(char *str, int n, FILE *stream);

Читает до перевода строки или n-1 байт и к концу строки присоединяет 0 байт, если прочитан \n.

```
void main(void){
  FILE*in;
  char string[80];
  in=fopen("story", "r");
  while(fgets(string,80, in)!=0)
     puts(string);
  }
```

Считывается до конца строки '\n' или 80-1 байт. При встрече EOF возвращает 0.

2. Запись текстовой строки в файл

int fputs(char *str,FILE *stream);

y=fputs(“Это строка”,in);

y-целое число, которое устанавливается в EOF, если fputs() встре-чает EOF или ошибку. fputs не добавляет '\n' в конец строки.

Все эти функции работают с текстовыми файлами и называются функциями последовательного доступа к файлу. Указатель внутри файла перемещается автоматически при чтении или записи.

Существуют функции прямого доступа к файлу.

Функции управления указателем в файле

Функция позволяет работать с файлом как с массивом. Достигать любого байта.

int fseek(FILE *stream, смещение, start)

Возвращает число типа int:

0 - если все хорошо;

-1 - ошибка.

Смещение – это количество байт на которое нужно сместить указатель по файлу с +(вперед), -(назад);

start - код начальной точки:

SEEK_SET или 0 – от начала файла;

SEEK_END или 2 – от конца файла;

SEEK_CUR или 1 – от текущего положения курсора.

fseek(in,0,0) - установить курсор на начало файла.

long int ftell(FILE *stream)- возвращает текущее положение курсора в файле.

Ввод/вывод записей фиксированной длины

Под записью фиксированной длины можно понимать размер элемента массива или структуры.

1. Чтение данных из двоичного файла.

int fread(void *ptr, size type, size n, FILE *stream)

void *ptr – адрес массива, куда записываются данные;

size type – размер типа в байтах;

size n – количество данных;

FILE *stream – указатель на файл.

```
void main(void){
  struct STOK record;
   FILE *in;
   in=fopen("data", "r");
   int n=fread(&record, sizeof(record), 1, in);
  }
```

Возвращает число считанных записей или EOF.

```
void main(void){
  float mas[100];
  FILE *in;
   In=fopen("data", "rb");
   fread(mas, sizeof(mas), 1, in);
  }        //можно так - fread(mas, sizeof(float), 100, in);
```

2. запись данных в двоичный файл.

int fwrite(void *ptr, size type, size n, FILE *stream)

Возвращает число записанных байт.
void *ptr – адрес массива, куда записываются данные;
size type – размер типа в байтах;
size n – количество данных;
FILE *stream – указатель на файл.

```
fwrite(mas, sizeof(mas), 1, in);
```

Пример 1. Запись во временный файл и чтение из него в массив.

```
#include <stdio.h>
#include <stdlib.h>
void main(void) {
int array[100];
        //создать временный файл
        FILE *tempf=tmpfile();
        if(!tempf) {
```

```
            puts("нельзя открыть временный файл");
            exit(1);
        }
        for(int index=0; index<100; index++)//пишем в файл
            fwrite(array,sizeof(int),1,tempf);
        rewind(tempf);          //указатель вернуть на начало
        fread(array,sizeof(int),100,tempf);
        rmtmp();                //закрыть и уничтожить временный файл
}
```

Пример 2. Проверить конец файлового потока

```
void main(void) {
int buff[100];
        FILE *fp;
        fp=fopen("prog.txt","r");
        if(!fp) {
            puts("нельзя открыть файл");
        }
        else {
            while(!feof(fp))
                if(fgets(buff,100,fp)!=NULL)
                    fputs(buff,stdout);
            fclose(fp);
        }
}
```

Пример 3. Создание копии файла

```
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
        FILE *in,*out;
        char c;
        fopen("a.ddd","r");
        fopen("b.ddd","r");
        fread(&c,1,1,in)
        while(!feof(in))
            {
            fwrite(&c,1,1,out);
            fread(&c,1,1,in)
            }
            fcloseall();
}
```